NPS67-80-013

NAVAL POSTGRADUATE SCHOOL

Monterey, California

SOFTWARE SYSTEM SAFETY

Donald M. Layton

October 1980

Final Report for Period Ending

September 1980

Approved for public release; Distribution unlimited

Prepared for:
Chief of Naval Research
Arlington, VA 22217

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral J. J. Ekelund
Superintendent

David A. Shrady
Acting Provost

The work reported herein was supported in part by the Foundation Research Program of the Naval Postgraduate School with funds provided by the Chief of Naval Research.

This report was prepared by:

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NPS67-80-013	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SOFTWARE SYSTEM SAFETY		5. TYPE OF REPORT & PERIOD COVERED FINAL REPORT FY 1980
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Donald M. Layton		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61152N; RR 000-01-10 N0001480WR00054
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE 14 October 1980
		13. NUMBER OF PAGES 25
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

system safety
safety
software analysis

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

An examination of software system safety analysis has been made and generalized techniques examined. These techniques parallel the techniques used for hardware analysis and are, in fact, predicated on the fact that the only safety perturbation in software is one that directs or misdirects a hardware component. Discussion is presented for a top to bottom and a bottom up hierarchial analysis, as well as an integrated technique.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT

An examination of software system safety analysis has been made and generalized techniques examined. These techniques parallel the techniques used for hardware analysis and are, in fact, predicted on the fact that the only safety perturbation in software is one that directs or misdirects a hardware component. Discussion is presented for a top to bottom and a bottom up hierarchical analysis, as well as an integrated technique.

I. BACKGROUND

Although the safety of the product has always received some consideration, albeit possibly tacit, the formal, systematic safety programs as we know them today did not come into being until the early 1960's. The one exception to this was the very strict safety controls established by the Atomic Energy Commission on the use and exposure to nuclear materials.

The first system safety documentation requirement was the United States Air Force Ballistic Systems Division Exhibit 62-41, "System Safety Engineering for the Development of Air Force Ballistic Missiles", published in April 1962. This document established System Safety requirements for the Associate Contractors on the Minuteman missile program.

In September 1963, the United States Air Force Specification, MIL-S-38130 (USAF), "General Requirements for Safety Engineering of Systems and Associated Subsystems and Equipment" was promulgated as the first military requirement for the engineering safety of general systems. This specification was closely followed in October 1963 by the Navy's similar (and nearly duplicate) requirement, MIL-S-38130(WEPS). These two documents were later merged into a joint specification, MIL-S-38130(ASG), and in June 1966, this specification became a Department of Defense (DoD) requirement, MIL-S-38130A.

In July 1969, the System Safety Specification was revised into a Military Standard, MIL-STD-882 and in July 1977, MIL-STD-882A expanded and defined in more specific terms the System Safety Program requirements.

In December 1978, DoD Instruction 5000.36, "System Safety Engineering and Management" stated that: "The Heads of DoD components shall establish system safety programs and apply Military Standard 882A....for each major system acquisition of other systems and facilities, as appropriate, based on the severity of associated hazards and the potential for loss or damage...".

The purpose of the System Safety Program, as stated in MIL-STD 882A is "To provide uniform requirements for developing and implementing a system safety program of sufficient comprehensiveness to identify the hazards of a system and to ensure that adequate measures are taken to eliminate or control the hazard".

The general requirements of the System Safety program are that safety, consistent with mission requirements, is built into the system in a timely, cost effective manner; that hazards associated with each system are identified, evaluated, and eliminated or controlled to an acceptable level throughout the entire life cycle of the system; and that retrofit actions required to improve safety are minimized through the timely inclusion of safety features during the development and acquisition of a system.

Each of the above listed requirements deserves special attention and comment. For example, it is to be noted that the Military Standard calls for safety consistent with mission requirement, not "safety at any cost". The utility of the system in the expected use environment is still the prime factor for consideration, and the design safety is intended, not to inhibit the mission, but rather to enhance the accomplishment of the mission.

It is also stated that hazards are to be identified, evaluated and eliminated or controlled to an acceptable level. This is the essence of what might be called the System Safety Process. The use of historical data, simulation, synthesis and test and evaluation are required to identify hazards of the system while the system is still in the design process.

The goal of this action is to minimize the risk, based on hazard severity, hazard probability, criticality, cost, time, resources and mission, throughout the lift cycle including disposition and disposal.

By inserting the safety process as early as possible in the design (and even concept) process, costly retrofit actions are reduced at a great savings of money, operational usage and limited resources. This takes safety from its prior "Band-Aid" approach of "try it and then fix it" into a new regime of designing the safety into the original product.

II. RISK ASSESSMENT

The general requirements of MIL-STD 882A include the risk assessment procedures of conducting hazard analyses starting in the Conceptual Phase and proceeding through the production phase. These analyses include:

1. Preliminary Hazard Analysis - a "broad brush" look at the potential hazards of systems and subsystems. The Preliminary Hazard Analysis (PHA) is begun long before detailed designs begin to take shape, and, although quite qualitative in nature, the PHA provides a base for other types of analyses.

2. Subsystem Hazard Analysis - Once detailed designs are underway, more in-depth analyses can be conducted on the subsystems. The subsystem may be a single component or part, or it may be a complex mini-system. Frequently the Subsystem Hazard Analysis (SSHA) is conducted on a fairly large block, and only if the analysis indicates a high degree of criticality is the analysis extended to lower level components. This is necessary due to the time and cost involved in the analysis procedure.

3. System Hazard Analysis - Once an SSHA has been completed, one has some degree of confidence that the critical hazards have been identified and eliminated or controlled to an acceptable level in each of the applicable subsystems. At this point a Systems Hazard Analysis (SHA) is conducted to investigate the safety of subsystem

interfaces. It is not at all unlikely that the interrelationships between two "safe" subsystems may result in some unsafe action of the combined system.

4. Operating and Support Analysis - Although it is easy to forget during the design/production process that the ultimate purpose is not the development of a system, but is rather the operational utilization of the system, one must keep in mind throughout the design/production phases that this system must be operated and supported to meet its mission utility. This is the purpose of Operating and Support Hazard Analysis (O&SHA) where, for the first time, the human element is injected into the equation. The O&SHA considers operation, support, maintenance, transportation and other operational uses of the system.

The purpose of all these analyses is to (a) identify any potential hazards, (b) evaluate the hazards, (c) assess the risk of the hazard, and (d) provide the necessary information required for the elimination or control of the hazard if the hazard is determined to be critical.

III. HAZARD ANALYSIS TECHNIQUES

Although the techniques to be used in conducting hazard analyses are generally at the option of the contractor, several analysis techniques that may be used are spelled out in MIL-STD 882A. These are:

1. Fault Tree Analysis - The Fault Tree is based on the Logic Tree procedure as developed by the Bell Laboratories. It was originally modified by the Boeing Company to trace fault developments in the Minuteman missile system. The Tree uses logic "gates" to build downward from a Head or Undesired Event. Inasmuch as a form of the Fault Tree Analysis may be used in Software analysis, some detail of this technique is in order. Consider a system whose wiring diagram[1] is shown below:

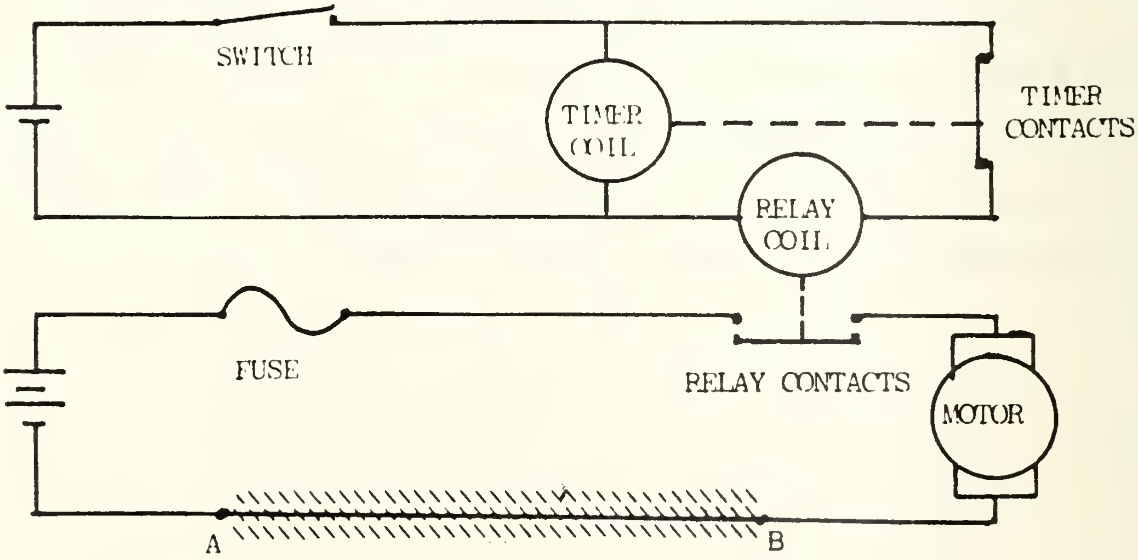

Figure 1. Wiring diagram for Fault Tree

[1] "Advanced Concepts in Fault Tree Analysis" by David Haasl. Paper presented at System Safety Symposium, Seattle, WA (1965).

When the Switch is closed, the Timer Coil is energized closing the Timer Contacts which puts power on the Relay Coil. With the Relay Coil energized, the relay contacts close, permitting current flow to the motor. It has been determined that overheating of the wire from point A to point B is critical to the system operation, and a Fault Tree is constructed with the Head Event "Overheated Wire".

Examination of the system shows that two events may produce an overheated wire, "Excessive current in the motor system wiring" and "Power applied to the system for an extended time". It is also seen that both of these occurrences must happen in concert. In other words, the "Overheated Wire" is a result of "Excessive Current" AND "Power Applied for an Extended Time". The logic event is, therefore, an AND gate.

To determine why the power may be applied for an extended time, we examine the circuit and note that this may occur if the power is not removed from the relay coil for some reason, or if the relay contacts fail in the closed position. We now have the next branch of our Tree with an OR gate connecting "Power not removed from relay coil" and "Relay contacts fail closed".

Similar development takes place proceeding downward from the Head Event until one reaches primary or secondary failures. Primary failures are those that occur while the part or component is operating within the parameters for which it was designed, and a Secondary failure is then the component is subjected to abnormal, out-of-design stresses.

A complete Fault Tree diagram for the system under consideration is shown in Figure 2.

Analysis of a Fault Tree may be either qualitative or quantitative. If the failure probabilities for each of the "end faults" of Figure 2 are known, the failure probabilities of each of the intermediate branches as well as the failure probability of the Head Event may be computed using Boolean Algebra.

But even a qualitative examination of many Fault Trees will provide useful information. With an AND gate leading directly to the Head Event, it is seen that inasmuch as both of the next lower level events must occur for the undesired Head Event to occur, eliminating either of the next level events will eliminate the failure of the Head Event.

Since the gate leading to the "Power applied for an extended time" is an OR gate, failure of either of the next lower level events will still produce this undesired subevent. This means that both of the next lower events must be eliminated to prevent this sub-failure.

On the other hand, the gate leading to "Excessive current in the system wiring" is also an AND gate, and this sub-failure will only occur if both of its sub-level events occur. One of these events, "Motor failed shorted" will occur only with a primary motor failure, leading us to the

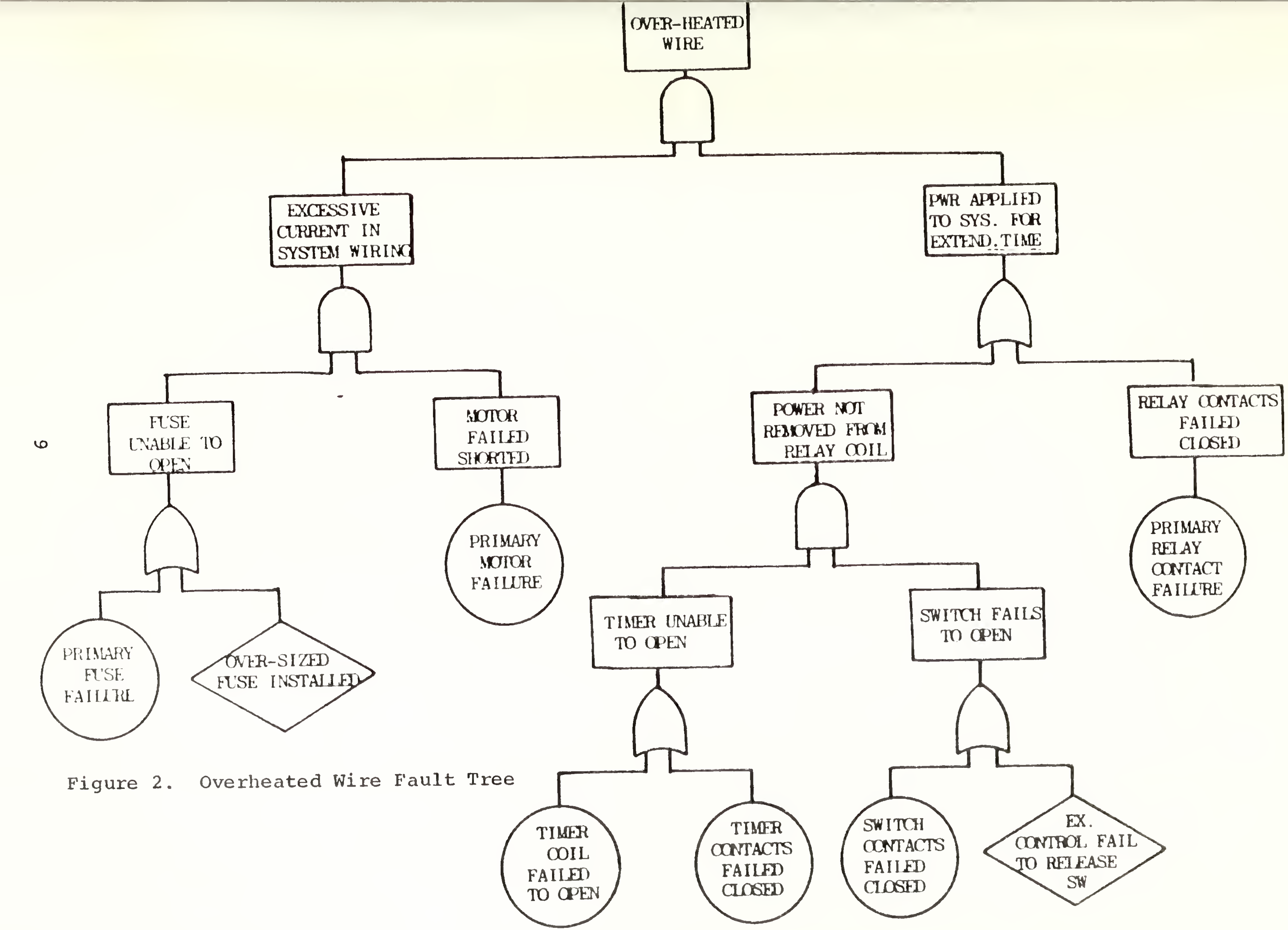
OVER-HEATED WIRE
EXCESSIVE CURRENT IN SYSTEM WIRING
PWR APPLIED TO SYS. FOR EXTEND. TIME
FUSE UNABLE TO OPEN
MOTOR FAILED SHORTED
POWER NOT REMOVED FROM RELAY COIL
RELAY CONTACTS FAILED CLOSED
PRIMARY MOTOR FAILURE
PRIMARY RELAY CONTACT FAILURE
PRIMARY FUSE FAILURE
OVER-SIZED FUSE INSTALLED
TIMER UNABLE TO OPEN
SWITCH FAILS TO OPEN
TIMER COIL FAILED TO OPEN
TIMER CONTACTS FAILED CLOSED
SWITCH CONTACTS FAILED CLOSED
EX. CONTROL FAIL TO RELEASE SW

Figure 2. Overheated Wire Fault Tree

conclusion that a high quality, high reliability motor will prevent (or at least reduce the probability of the occurrence of) the Head Event.

We see also that if we can prevent the fuse from failing to open, we can also prevent excessive wiring. This may be accomplished by both preventing the insertion of an oversized fuse (design of the fuse holder) and prevention of a primary fuse failure in which the fuse fails to open with excessive current. With most fuse designs, this type of fuse failure should be extremely remote.

2. Fault Hazard Analysis - A second technique cited in MIL-STD 882A is the Fault Hazard Analysis (FHA), which is based on the techniques long in use by Reliability called Failure Mode and Effect Analysis (FMEA) or Failure Mode, Effect, and Criticality Analysis (FMECA). The Fault Hazard Analysis, unlike the Fault Tree Analysis starts at the bottom and works up, rather than at the top working down. Whereas, in the Fault Tree Analysis we conducted our analysis to determine what failures would cause an undesired event, in the Fault Hazard Analysis, one starts with the part or component failure and determines the ultimate effect of that failure.

The Fault Hazard Analysis may be used in detailed examination, such as in an SHA where one is considering failure modes and effects of detailed designs, or it may be used in a less structured format in an overall analysis of design concepts, as in the Preliminary Hazard Analysis.

As with the Fault Tree Analysis, the Fault Hazard Analysis may be either qualitative or quantitative. A qualitative analysis usually precedes any quantification.

3. Sneak Circuit Analysis - A third technique referenced in MIL-STD 882A is that of the Sneak Circuit Analysis. This is a technique, originally developed by the Boeing Aerospace Company, to locate hazards that might occur without a failure in the system. A classic example of a Sneak Circuit was in a 1960's imported car in which the radio and the brake lights both received their power from a common terminal on the switch. The brake lights were also capable of being powered through the emergency flasher module, and this portion of the wiring was such that when the brake pedal was depressed, even with the ignition switch off, the brake lights were illuminated - and the radio came on!

In addition to these three techniques cited in MIL-STD 882A, there are many others, lesser known analysis techniques. These include Energy Transfer techniques wherein all subsystems and systems are considered on the basis of energy in - energy out, and Resource Utilization techniques that consider desired (and undesired) outputs codified on the basis of input resources.

IV. SOFTWARE ANALYSIS - GENERAL

Software reliability predictions are available that provide numerical predictions of how many errors will remain in the software when it is delivered, but these predictions do not tell the effects of a software error, where it will occur, or in what mission phase.

Several analysis techniques have been used to give partial answers to the software system safety questions, but these have been, in general, limited to individual disciplines and/or to specific categories of failure.

There is, however, one factor that acts as a catalyst for the solution of the software system safety problem, and that is the fact that any software difficulty is translatable into a general system safety problem only if there is a hardware involvement. That is to say, the only software 'mishap" that has safety implication is one that commands a hardware function at the wrong time, in the wrong sequence, in the wrong manner or when the hardware component should not be commanded at all.

This fact, in itself, gives rise to possible software system safety techniques in which the software 'mishap' is directly related to hardware 'mishaps' for which there are well known and experienced analysis techniques, as discussed in Part III. It would appear, therefore, that an examination of each known hardware mishap for possible software command functions would suffice as a software system safety analysis. And indeed it would, except that this wholesale approach would be extremely time consuming and, as a result, extremely costly. One would also have to exercise great caution to ensure that software interfaces are considered in addition to the already considered hardware interfaces.

Although there are many proponents of various single technique methods for software system safety analysis, even these techniques are generally modified so as to take advantage of other techniques in some degree to reduce the cost and time that would be otherwise involved.

In general, software analysis techniques for system safety follow the hardware analysis methods. One method is the 'bottom up' method similar to the Failure Mode and Effect Analysis and another is the 'top down' procedure related to the Fault Tree Analysis. Some detailed examination of these two methods, as well as a combined method will be discussed in subsequent parts of this summary.

V. TOP DOWN SOFTWARE ANALYSIS

The Top Down analysis technique is based on the determination of hazardous termination points for the software program. This is analogous to the Undesired Event which is the Head Event of the hardware Fault Tree Analysis.

The first step in the Top Down techniques is to define Acceptable Terminations as well as Hazardous Terminations. Such a definition list, as related to a missile system, is as follows:[1]

<u>Acceptable Defined Terminations</u>

1. Missile successfully launched.
2. Missile aborted with booster and warhead safe.
3. Missile aborted with booster or warhead not saved and with operator alerts.
4. System cycled to Hold with operator alerts.
5. System cycled to Test Mode with operator alerts.
6. System recycled to known states with operator alerts.
7. System automatically cycled to power down with appropriate operator alerts.

<u>Hazardous Terminations</u>

1. Unauthorized launch of a missile.
2. Unintentional launch of a missile, including the launch of a wrong missile.

[1] Software Safety and Security Analysis Techniques and Methods, Vitro Laboratories, Silver Spring, MD, 1980.

3. Missile abort without operator alerts. The alerts to include the safety evaluation of the aborted missile.
4. A premature issuance of a unique pre-arm signal.
5. An abort without adequate declassification.
6. An unauthorized disclosure of classified data.
7. Any unauthorized or unintentional modification of safety and/or security sensitive programs or data.
8. Any termination which is not a member of the set of acceptably defined terminations.

It is to be observed that several of the Hazardous Termination categories go well beyond the safety of the hardware. For example, Item number 3, "Missile abort without operator alerts" might appear to have "Fail Safe" connotations in that the missile aborted, presumably due to proper software/hardware interaction. However, a new and unique safety problem may arise if the operator is unaware of the abort or of the reason for the abort and attempts to operate the missile again.

With the hierarchial Top Down development process, the software design cycles is as shown in Figure 3. As in the hardware system safety, the best payoff occurs if the software system safety is inserted early in the design process. Although it is true that many systems have an initial hardware design followed by the design of the control software, even these systems should undergo a software analysis for subsystems and for system interfaces as early in the design process as practicable.

For those subsystems and systems in which the software is "King", that is to say, for systems in which the software is the principal, and possibly the first, design item and then the hardware is designed to accomplish the desired outputs of the system, early analysis of the software is essential.

The flow of control, as shown in Figure 3, starts at the top level, goes down one or more levels, comes back as required, and then goes down to another level.

The actual assessment depends on the construction of a series of binary trees that have the following attributes:

1. There is a one-to-one correspondence between decision points of the software and nodal branch points of the tree.

2. Each branch can be categorized by a series of state transitions that can be described by Boolean alogorithms.

3. The origin of each binary tree is always an operation action, a machine-generated priority interrupt, a periodic status monitoring, or a return to a higher level process.

Figure 4 is a generalized Binary State Transition Tree which illustrates all constructs. Such a tree will be examined for possible occurrences of system errors caused by program errors (incomplete definition of state vectors or predicate

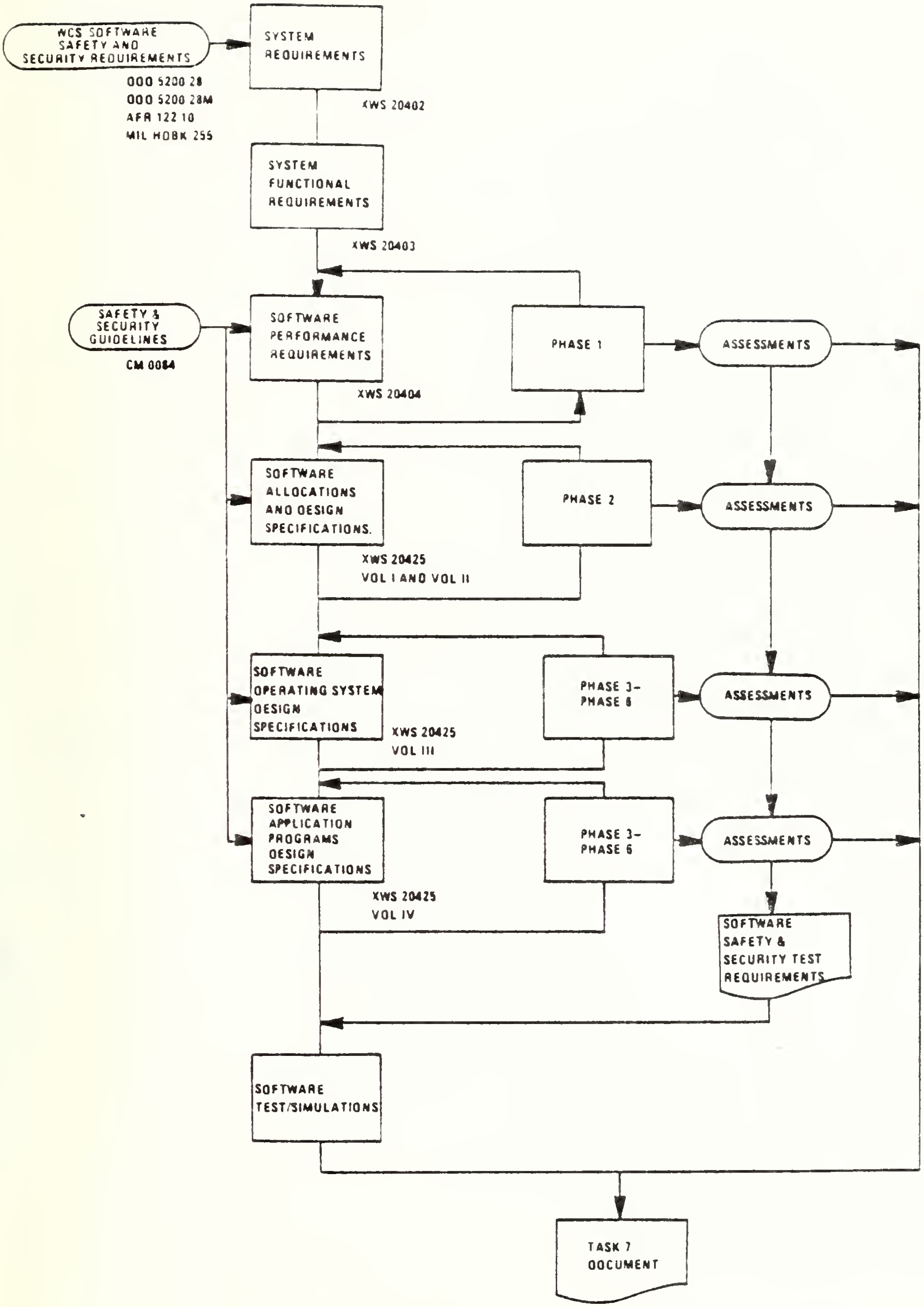
WCS SOFTWARE SAFETY AND SECURITY REQUIREMENTS
000 5200 28
000 5200 28M
AFR 122 10
MIL HDBK 255
SYSTEM REQUIREMENTS
XWS 20402
SYSTEM FUNCTIONAL REQUIREMENTS
XWS 20403
SAFETY & SECURITY GUIDELINES
CM 0084
SOFTWARE PERFORMANCE REQUIREMENTS
XWS 20404
PHASE 1
ASSESSMENTS
SOFTWARE ALLOCATIONS AND DESIGN SPECIFICATIONS.
XWS 20425
VOL I AND VOL II
PHASE 2
ASSESSMENTS
SOFTWARE OPERATING SYSTEM DESIGN SPECIFICATIONS
XWS 20425
VOL III
PHASE 3-PHASE 6
ASSESSMENTS
SOFTWARE APPLICATION PROGRAMS DESIGN SPECIFICATIONS
XWS 20425
VOL IV
PHASE 3-PHASE 6
ASSESSMENTS
SOFTWARE SAFETY & SECURITY TEST REQUIREMENTS
SOFTWARE TEST/SIMULATIONS
TASK 7 DOCUMENT

Figure 3. Relationship between Software Analysis and Software Design

transforms) or by single or multiple hardware failures. System errors are treated as virtual branch points in which the occurrence and subsequent processing is also treated as a binary tree.

Such a tree is called a Binary Fault Termination Tree and is also shown in Figure 4. This tree has end points that may describe hazardous terminations, and, as such, is of prime interest in software system safety analysis.

As has been previously stated, the only software 'mishap' that is of concern in the system safety analysis is one that produces an undesired hardware event. For this reason, software errors that occur but are trapped and contained and do not produce hazardous terminations are of but secondary interest in the safety analysis. The Binary Fault Termination Trees are used to evaluate whether checkpoints are established to trap and contain the software errors.

When errors have been identified and the processing shows that a hazardous termination is being approached, the process may be translated into a probability function to determine the probability of the hazardous termination. This translation implies that the following probabilities can be computed:

1. The probability that the system moves into the implied mode.

2. The probability that the system moving from the implied mode misses the illegal procedure trap.

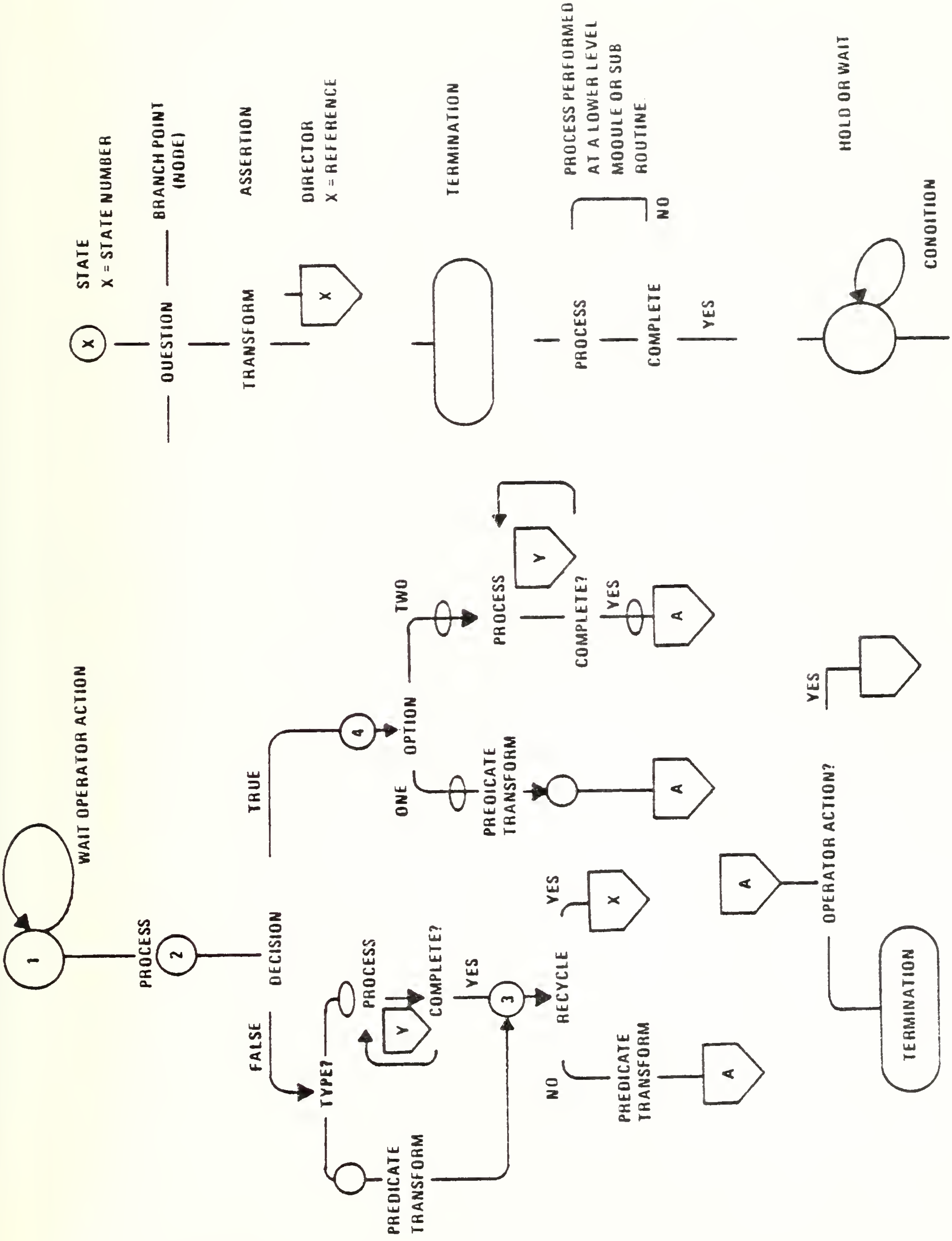

Figure 4. Binary State Transition Tree

VI. BOTTOM UP SOFTWARE ANALYSIS

Another technique that is used for software analysis is the Bottom Up Analysis. This technique is based on the predication that any hazard that is created or allowed to propogate must exist in hardware. Because of this, the software safety analysis should analyze, within the software boundaries, concerns identified within the hardware facets. This stipulates that the software analysis is an extension of the Preliminary Hazard Analysis (PHA).

Inasmuch as the Subsystem Hazard Analysis (SSHA) and the System Hazard Analysis (SHA) follow the Preliminary Hazard Analysis, the command and control software development and analysis should follow the basic development of the hardware that is to be controlled.

This offers the use of the techniques similar to the Fault Hazard Analyses of hardware system safety. This technique, which is akin to the long-used Reliability techniques of Failure Mode and Effect Analysis (FMEA) or the Failure Mode, Effect and Criticality Analysis (FMECA), considers the undesired outcomes of the failure of a subsystem or component.

The software analyses should address hazards resulting from basic deficiencies in the requirements, the software program design, the internal coding, the software testing, the user interfaces, the backup software functions and the hardware/software interfaces.

The general approach for this technique is as follows:[1]

1. Divide the system into portions to gain insight into sub-function operations.

2. Conduct the hardware system safety analysis.

3. Conduct the software system safety analysis.

a. Analyze for correct implementation of the hardware functional requirements and interfaces.

(1) Review the functional requirements of the subsystems being controlled.

(2) Analyze the requirements of the software definition and implementation to ensure that the software is kept in a safe configuration.

b. Analyze for internal software anomalies which would affect the execution of the software.

(1) Analyze those functions that have been designated safety critical.

(2) Analyze those functions that might affect the execution of the safety critical functions.

(3) Review for implementation of errors or anomalies in the detailed requirements, the detailed program design and/or the coding.

(4) Review the timing of competing/overlapping functions.

[1] Software Safety Analysis, A Presentation by John G. Griggs, Martin Marietta Corp. at Tri-Service Safety Conference Colorado Springs, CO, September 1980.

(5) Review the use of incorrect data and/or changing data.

(6) Review the use of coding techniques to minimize the effects of errors.

VII. COMBINED TECHNIQUE

A combination of the Top Down and the Bottom Up Techniques has been developed by the Boeing Aerospace Company[1] into a technique called Integrated Path Critical Analysis (ICPA).

The seven step ICPA combines Fault Tree Analysis for the identification of critical paths, Failure Mode and Effect Analysis for component failure rates, Sneak Circuit Analysis for actual detailed system configuration, Component Sensitivity Analysis for the determination of areas of emphasis and Critical Path Analysis to evaluate and quantify overall system reliability and safety.

The steps outlined by Boeing Aerospace for this technique are as follows:

1. Determine the Scope of the Analysis.

a. This is usually accomplished by reviewing the top level hazards and/or critical functions and then deciding which ones should be analyzed in detail and to what level. This may be done by using existing Fault Trees and FMEA's and marking them for particular emphasis.

2. Establishment of Accurate System Configuration in the form of Integrated Functional Network Trees.

[1] Software/Hardware Integrated Critical Path Analysis (ICPA) by J. H. Campbell and F. H. Tuna in Proceedings of Fourth International System Safety Conference, July 9-13, 1979.

a. It is essential that the analysis be conducted on an accurately represented system. The use of logic trees assists in the determination of the accuracy of the system configuration under examination.

3. Updating and Integrating Fault Trees.

a. Here one applies the material generated from Step #2 and adds the software network threats which change the impact of the hardware due to usage and/or interconnect hardware, thereby changing the hardware interrelationships.

4. Reliability Prediction of Critical Paths.

a. Once the critical paths have been defined, the reliability numbers can be affixed to present a reliability prediction.

5. Procedure Analysis.

a. This step considers all procedures including test, contingency and backup.

6. Failure Effect Analysis.

a. The previous steps have emphasized the possible critical steps and paths, and these functions are now analyzed.

7. Analyses Reports.

a. These reports are generated to permit management decisions in regard to changes and alterations to the suspect subsystems and systems. These reports should contain the revised Fault Trees, sensitivity statements, hardware/software interaction problems and the failure effects of critical paths.

VIII. CONCLUSIONS

The subject of software system safety is relatively new. Concern in the Electronic Industries Association (EIA) was first expressed about a year and a half ago and the EIA formed a task force to develop a generic approach to the analysis of software system safety in August 1979.

This action is significant due to the activity of the G-48 Committee of the EIA in developing codified System Safety requirements and fostering the development of System Safety analysis techniques.

There appear to be techniques available for the analysis of System Safety software problems, and these techniques are, in general, based on the proven techniques used in hardware System Safety analysis.

Despite the particular technique to be used, it appears that software analysis parallel with software development has the greatest payoff at this time.

Although the analysis techniques in greatest use to date have been, at the most, a modification of the hardware techniques, there are people working on different techniques which have, as yet, not demonstrated any spectacular results. This is not to say, however, that such methods are not to be closely monitored, because it is quite possible that these, rather radical, methods may yet have a significant payoff.

DISTRIBUTION

NO. OF COPIES

1. Library 2
   Code 0142
   Naval Postgraduate School
   Monterey, CA 93940

2. Research Administration 1
   Code 012A
   Naval Postgraduate School
   Monterey, CA 93940

3. Chief of Naval Research 2
   Arlington, VA 22217

4. Professor Donald M. Layton 5
   Code 67Ln
   Naval Postgraduate School
   Monterey, CA 93940

5. Defense Technical Information Center 2
   Cameron Station
   Alexandria,Virginia 22314